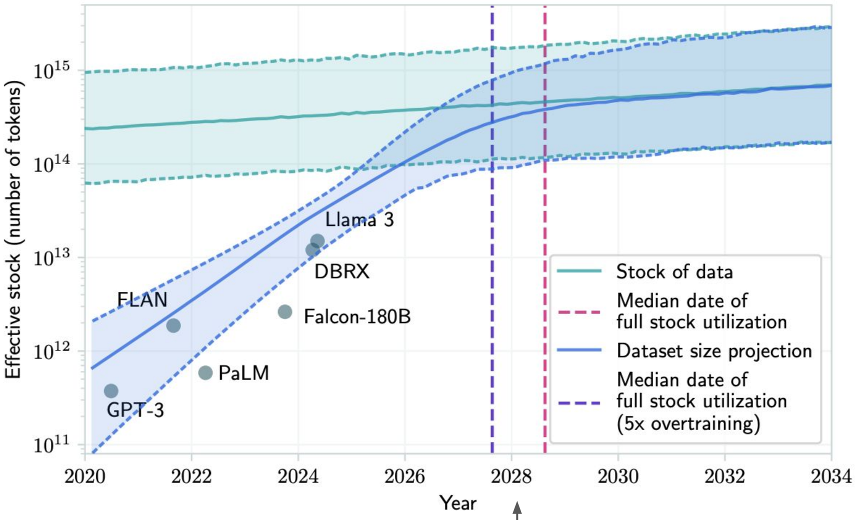
Data Attribution and Optimization for Data-Efficient AI

Xiaoqiang Lin

July 4th 2025 @ SJTU

Data Consumption VS Data Production

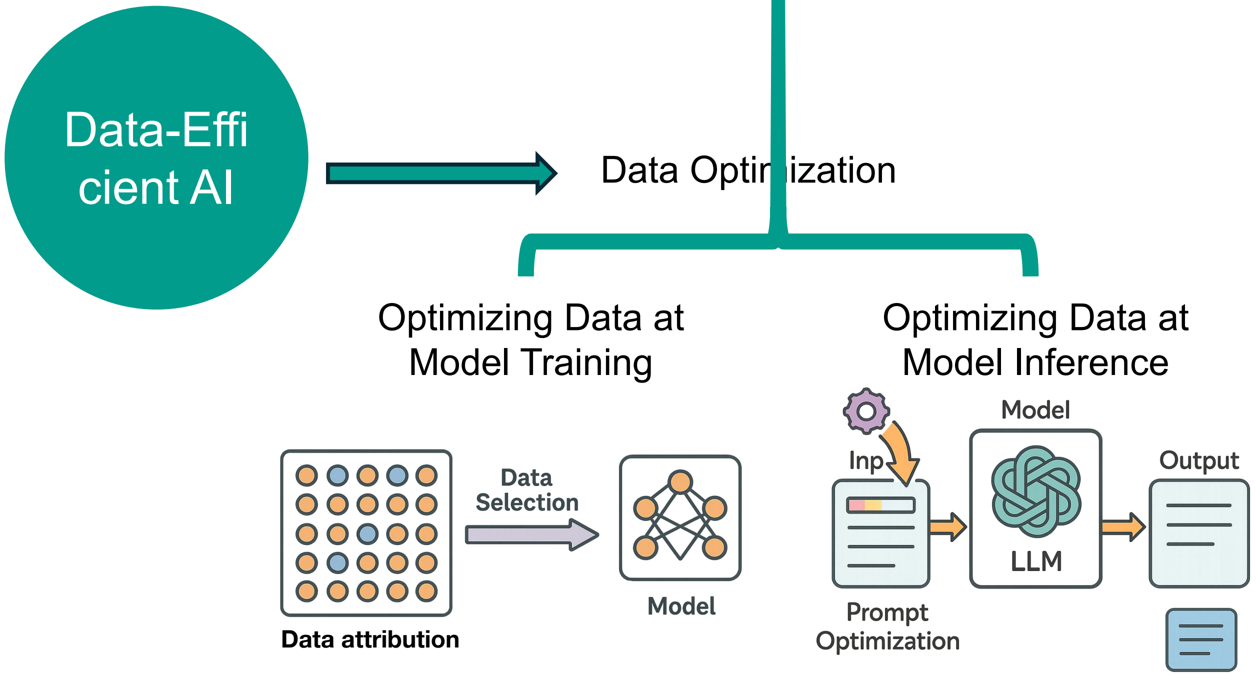


We will run out of data stock here! [1]

- Data consumption grows faster than production.
- Increase model does not improve performance proportionally.

[1] Villalobos, P., Ho, A., Sevilla, J., Besiroglu, T., Heim, L., & Hobbhahn, M. (2024, July). Position: Will we run out of data? Limits of LLM scaling based on human-generated data. ICML 2024.

Data-Efficient AI



Agenda

Optimizing data at model training:

- FreeShap: Data Attribution for LLM Fine-tuning, ICML 2024
- NICE: Data Attribution for Non-differentiable Metrics, ICML 2025

Optimizing data at model inference:

- INSTINCT: Black-box Prompt Optimization, ICML 2024
- POHF: Prompt Optimization with Human Feedback, ICML workshop 2024

FreeShap

Helpful or Harmful Data? Fine-tuning-free Shapley Attribution for Explaining Language Model Predictions.

Jingtan Wang*, Xiaoqiang Lin*, Rui Qiao*, Chuan-Sheng Foo, Bryan Kian Hsiang Low.

ICML 2024.

Motivation

The Data Value / Data Contribution

Marginal contribution of i for the whole dataset

The leave-one-out (LOO) value of the -th data point: (~influence function)

$$\xi_i = \boxed{U(N) - U(N \backslash \{i\})}$$

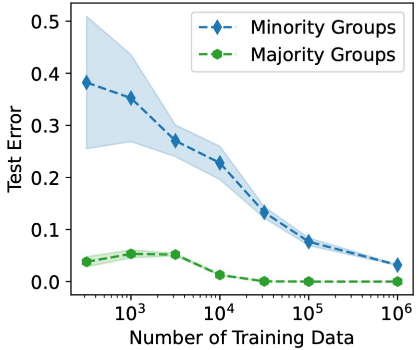
The Shapley value of the i-th data point:

$$\phi_i = \sum_{S \subseteq N \backslash \{i\}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \boxed{(U(S \cup \{i\}) - U(S))}$$

Marginal contribution of i for the subset

Preliminaries

- The harm of the mislabeled data is magnified when the dataset has a smaller size.

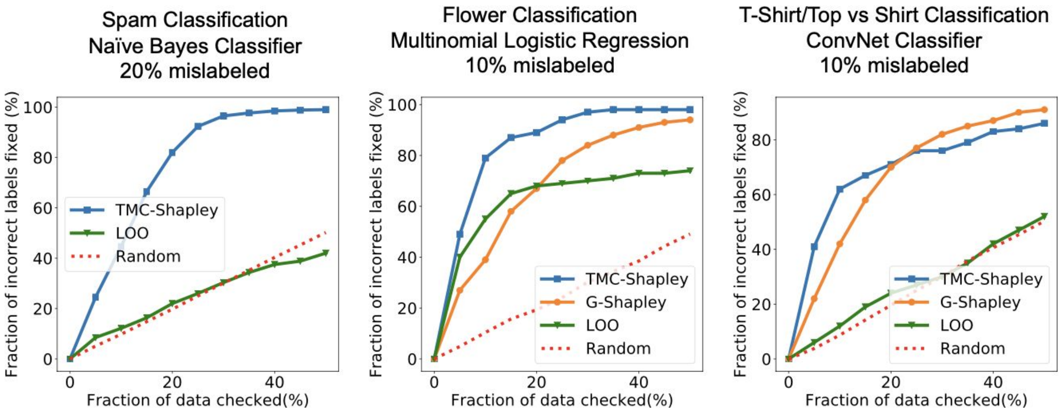


- The Shapley value is better at detecting mislabeled data because of the consideration of the smaller subsets.

Motivation: Shapley Value is More Effective

The Shapley Value

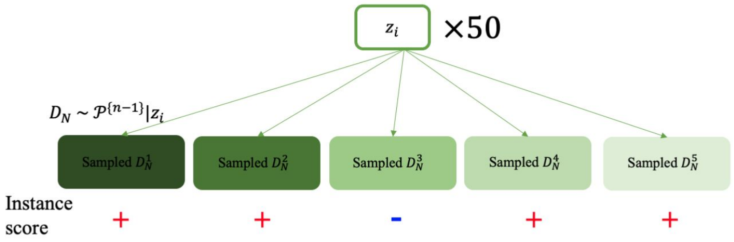
- The Shapley value is much better at identifying mislabeled data than LOO values



[Ghorbani and Zou, 2019]

Methodology: Shapley Value is More Robust

- **Place the same text sample into different training datasets and evaluate their instance scores**



- **Compared with LOO, Shapley value can output helpfulness/harmfulness consistently**
    - Usefulness in one dataset is generalizable to other datasets

Text example	Label	Shap	Shapley score	LOO	LOO score
confident filmmaking and a pair of fascinating performances	[pos]	5+/0-	0.45 ±0.23	3+/2-	0.23±1.52
seriously, rent the Disney version	[neg]	5+/0-	1.85±0.72	3+/2-	-0.70±2.36
it's not going to be everyone's bag of popcorn, but it definitely gives you something to chew on'	[pos]	0+/5-	-1.91±0.57	3+/2-	0.23±1.34
would fit chan like a $ 99 bargain-basement special	[neg]	0+/5-	-0.36±0.20	2+/3-	0.00±1.45

Challenge of SV in Large Models Training

Computational Scalability

$$\phi_i = \underbrace{\sum_{S \subseteq N \setminus \{i\}}}_{2^{|N|-1} \text{ number of calculations}} \frac{|S|!(|N| - |S| - 1)!}{|N|!} \underbrace{(U(S \cup \{i\}) - U(S))}_{\text{requires fine-tuning}}$$

- The naive evaluation of the utility function requires fine-tuning a (large) pretrained model on the subset, then evaluate the model's performance.
- Repeated fine-tuning are extremely costly.

Solution

The Empirical "Kernel" Trick

$$\Psi = \begin{bmatrix} \nabla_\theta f(x_1)^\top \\ \nabla_\theta f(x_2)^\top \\ \nabla_\theta f(x_3)^\top \\ \vdots \\ \nabla_\theta f(x_n)^\top \end{bmatrix}$$

empirical Neural Tangent Feature

$$K = \begin{pmatrix} \nabla_\theta f(x_1)^\top \nabla_\theta f(x_1) & \nabla_\theta f(x_1)^\top \nabla_\theta f(x_2) & \cdots \\ \nabla_\theta f(x_2)^\top \nabla_\theta f(x_1) & \nabla_\theta f(x_2)^\top \nabla_\theta f(x_2) & \cdots \\ \vdots & \vdots & \ddots \end{pmatrix}$$

empirical Neural Tangent Kernel (eNTK)

Prior work: Kernel regression on the eNTK resembles fine-tuning.

[Wei et al., 2022]

FreeShap

Fine-tuning-free Shapley Value

FreeShap amortizes the fine-tuning cost by pre-computing the eNTK matrix, then calculates the utility terms of the Shapley value using models obtained from kernel regressions.

FreeShap

Fine-tuning-free Shapley Value

$$\phi_i = \sum_{S \subseteq N \setminus \{i\}} \frac{|S|!(|N|-|S|-1)!}{|N|!} \left(\boxed{U(S \cup \{i\})} - \boxed{U(S)} \right)$$

a funny little film

a well-put-together piece

a shaky, uncertain film

$$\blacktriangleright \begin{bmatrix} \nabla_\theta f(x_1)^\top \\ \nabla_\theta f(x_2)^\top \\ \vdots \\ \nabla_\theta f(x_n)^\top \end{bmatrix} \blacktriangleright K = \begin{bmatrix} k_{11} & k_{12} & k_{13} & \cdots & k_{1n} \\ k_{21} & k_{22} & k_{23} & \cdots & k_{2n} \\ k_{31} & k_{32} & k_{33} & \cdots & k_{3n} \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ k_{n1} & k_{n2} & k_{n3} & \cdots & k_{nn} \end{bmatrix} \blacktriangleright$$

$$\left(\begin{bmatrix} k_{11} & k_{12} \\ k_{21} & k_{22} \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} y_1 \\ y_2 \end{bmatrix}$$

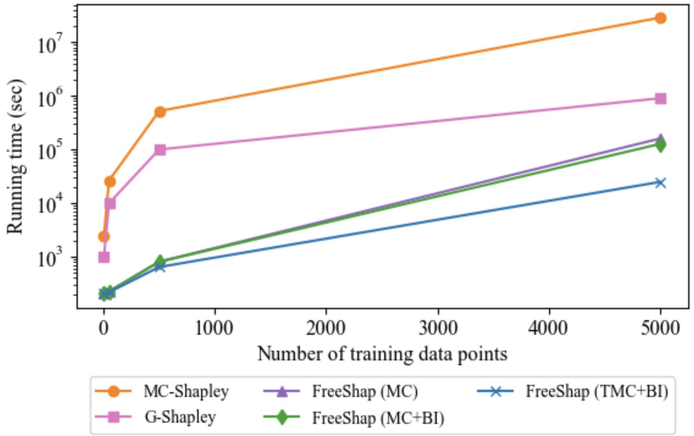
$$\left(\begin{bmatrix} k_{22} & k_{23} & k_{2n} \\ k_{32} & k_{33} & k_{3n} \\ k_{n2} & k_{n3} & k_{nn} \end{bmatrix} + \lambda I \right)^{-1} \begin{bmatrix} y_2 \\ y_3 \\ y_n \end{bmatrix}$$

Input	Gradient	empirical NTK	Kernel Regression on Submatrices
x_i	$\psi(x_i) = \nabla_\theta f(x_i)^\top$	$K, K_{ij} = \psi(x_i)\psi(x_j)^\top$	$(K[S,S] + \lambda I_{\lvert S \rvert})^{-1} \mathbf{y}_S$

FreeShap

Efficiency

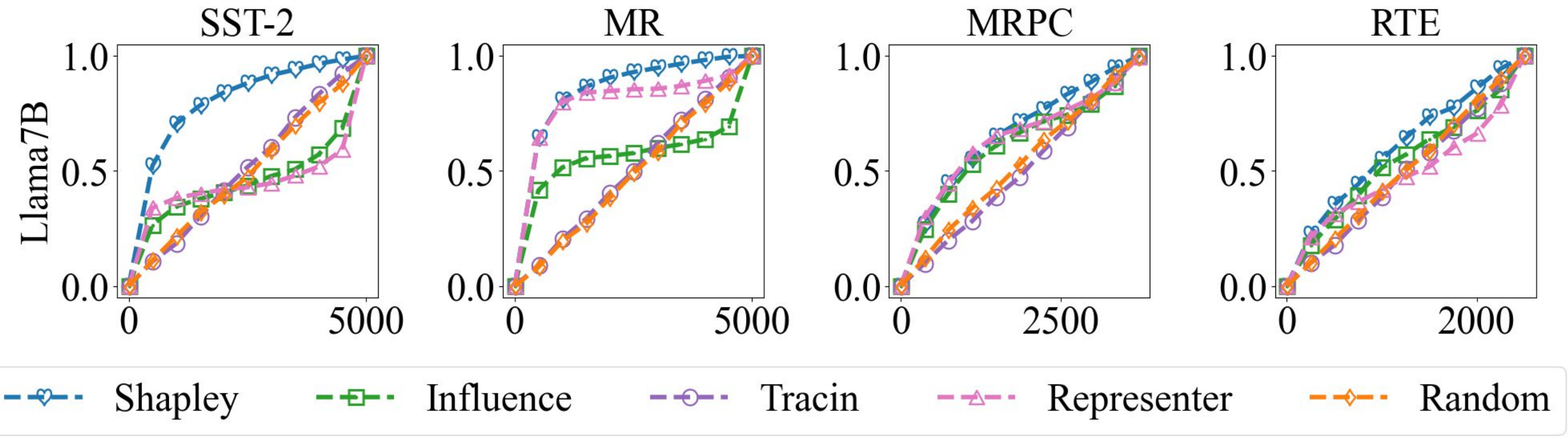
- FreeShap is significantly faster than other approximated Shapley value baselines.



Dataset	SST-2
Model	BERT

Data Curation: Wrong Label Detection

- Poison training set by flipping 10% of data, and then examine the poisoned data by reviewing data points in the order of their scores from lowest to highest.
- Shapley **excels in detecting mislabeled data** within datasets.



Data Selection

- Setting: A train set, a test set for calculating training point scores, and a *held-out* set evaluating the selected subsets.
- Sequentially add training data points with the highest scores.
- **The higher the performance increase, the better the data curation approach is**.
- Shapley value is also effective for data curation when test distribution is unknown.

BERT	2%	4%	6%	8%	10%
Shapley	0.1951	**0.2111**	**0.2148**	**0.2167**	**0.2223**
Influence	0.0272	0.0647	0.0393	0.0647	0.0750
TracIn	0.1370	0.1904	0.2008	0.1548	0.1970
Representer	0.1182	0.1351	0.1388	0.0619	0.1238
Random	**0.1970**	0.1951	0.2073	0.2017	0.1529

Llama2	2%	4%	6%	8%	10%
Shapley	**0.0816**	**0.1398**	**0.1754**	**0.2186**	**0.2458**
Influence	-0.0038	-0.0038	-0.0038	-0.0038	-0.0038
TracIn	0.0638	0.1144	0.1304	0.1773	0.2092
Representer	-0.0019	0.0000	0.0019	0.0066	0.0056
Random	0.0638	0.1041	0.1529	0.1792	0.1951

Summary

- FreeShap provides an efficient and scalable approximation of the Shapley value.
- FreeShap demonstrates strong capability in mislabeled data detection, advancing data diagnostics, which can be used to increase model reliability.

NICE

NICE: Non-differentiable Evaluation Metric-based Data Selection for Instruction Tuning.

Jingtan Wang, Xiaoqiang Lin, Rui Qiao, Pang Wei Koh, Chuan-Sheng Foo, Bryan Kian Hsiang Low.

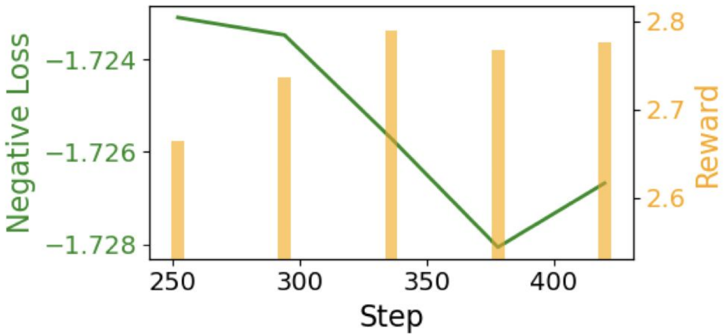
ICML 2025.

Motivation

Loss-based influence (TracIn, Influence function, etc.) estimates the effect of each training data on the validation loss via the gradient of the validation loss.

- Discrepancy Between Loss and Evaluation Metrics
- Reliance on Labeled Validation Data

$$\mathcal{I}_{\text{up,loss}}(z, z_{\text{test}}) \stackrel{\text{def}}{=} \frac{dL(z_{\text{test}}, \hat{\theta}_{\epsilon,z})}{d\epsilon}\bigg|_{\epsilon=0} = \nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top \frac{d\hat{\theta}_{\epsilon,z}}{d\epsilon}\bigg|_{\epsilon=0} = -\nabla_\theta L(z_{\text{test}}, \hat{\theta})^\top H_{\hat{\theta}}^{-1} \nabla_\theta L(z, \hat{\theta}).$$



NICE: Non-differentiable evaluation metrics-based InfluenCe Estimation

Instead of using validation loss, we use $$L_r(z_v;\theta) = \mathbb{E}_{\hat{y}_v \sim f(y|x_v;\theta)}[-r(z_v,\hat{y}_v)]\ ,$$

Quantify the influence of including a training point z_i on the a validation point z_v's performance measured by a non-differentiable evaluation metric **r**

- Code pass rate, math correct rate
- Reward model

TracIn

Training point's gradients w.r.t. loss

$$Inf_{NICE} = \sum_{e=1}^{E} \boxed{\bar{\eta}_e} \langle \boxed{\nabla_\theta L(z_i;\theta^e)}, \mathbb{E}_{\hat{y}_v \sim f(y|x_v;\theta^e)}[-\nabla_\theta \log(f(\hat{y}_v|x_v;\theta^e))r(z_v,\hat{y}_v)] \rangle$$

across checkpoints

Validation point's **policy gradients** w.r.t. eval metrics

Influence function

$$Inf_{NICE_{IF}} = \boxed{\mathbb{E}_{\hat{y}_v \sim f(y|x_v;\theta^E)}[-\nabla_\theta \log(f(\hat{y}_v|x_v;\theta^E))\boxed{r(z_v,\hat{y}_v)}]}^T \boxed{H_{\theta^E}^{-1}} \boxed{\nabla_\theta L(z_i;\theta^E)}$$

computed w.r.t final checkpoints

Training point's gradients and Hessians w.r.t. loss

Validation point's **policy gradients** w.r.t. eval metrics

NICE: Non-differentiable evaluation metrics-based InfluenCe Estimation

Approximated via Monte-Carlo Sampling

Training point's gradients w.r.t. loss

$$Inf_{NICE} = \sum_{e=1}^{E} \boxed{\bar{\eta}_e} \langle \boxed{\nabla_\theta L(z_i; \theta^e)}, \mathbb{E}_{\hat{y}_v \sim f(y|x_v;\theta^e)}\left[-\nabla_\theta \log\left(f(\hat{y}_v|x_v;\theta^e)\right) r(z_v, \hat{y}_v)\right]\rangle$$

across checkpoints

Validation point's **policy gradients** w.r.t. eval metrics

NICE: Non-differentiable evaluation metrics-based InfluenCe Estimation

$$Inf_{NICE_{IF}} = \boxed{\mathbb{E}_{\hat{y}_v \sim f(y|x_v;\theta^E)}\left[-\nabla_\theta \log\left(f(\hat{y}_v|x_v;\theta^E)\right)\boxed{r(z_v,\hat{y}_v)}\right]}^T \boxed{H_{\theta^E}^{-1}} \boxed{\nabla_\theta L(z_i;\theta^E)}$$

computed w.r.t final checkpoints

Training point's gradients and Hessians w.r.t. loss

Validation point's **policy gradients** w.r.t. eval metrics

- NICE Preferred training points: Those whose gradient are most similar to the policy gradients of the validation performance measured by the reward function r, i.e., those training points can improve the validation performance more
- 💡 When r does not require the the ground truth label y (reward model which only needs prompts x_y and generated response \hat{y}_v) ➡️ **NICE can output score without need of validation label!**

Generalization to Other Loss-based Influence Estimation Methods:

Method	AlpacaEval	TLDR	RLHF	HumanEval
IF (DataInf)	11.11	2.01	0.83	37.40
NICEIF	**20.44**	**3.97**	**1.89**	**39.68**

NICEIF consistently outperforms Vanilla Influence Function

Assisted Monte Carlo

Benefit of Monte-Carlo Sampling used in NICE: Utilizing multiple different responses, offering diverse guidance. The generated response can be better than the label response

Limitation: When the model is too weak, the MC samples may not contain high-quality responses with high rewards

Solution: Assisted Monte Carlo (AMC) uses a model that is better at the target task to assist generation

Assisted generated responses

$$Inf_{NICE_{AMC}} = \sum_{e=1}^{E} \bar{\eta}_e \left\langle \nabla_\theta L(z_i; \theta^e), \mathbb{E}_{\boxed{\hat{y}_v^* \sim g(y|x_v;\psi)}}\left[-\nabla_\theta \log\left(f\left(\boxed{\hat{y}_v^*}|x_v; \theta^e\right)\right) r\left(z_v, \boxed{\hat{y}_v^*}\right)\right]\right\rangle$$

Experimental Results

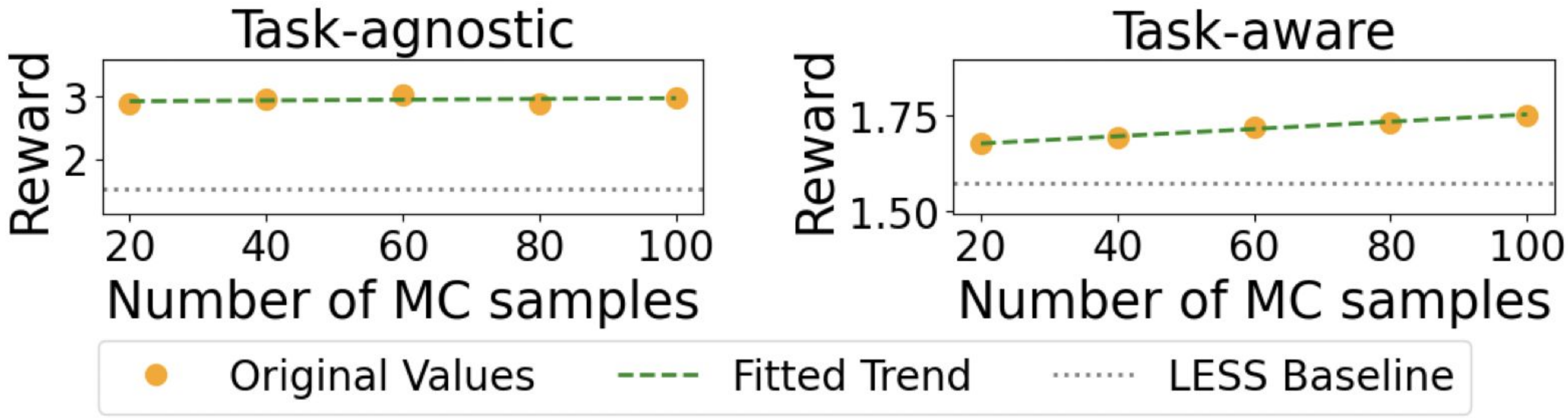
Major Findings:

- NICE outperforms the loss-based influence estimation.
- No labels? No problem! NICE outperform baselines that utilize the label response.
- Less is more: subset outperforms the full dataset.
- Assisted monte-carlo sampling can boost data selection when the size of training data is large (task-agnostic setting).

Model & Dataset		Full	Random	RDS	BM25	DSIR	TSDS	LESS	NICE	NICEAMC
Task-agnostic										
Llama2-7B	AlpacaEval	22.59	16.13$_{\pm 1.18}$	14.70	19.60	20.27	17.40$_{\pm 2.44}$	26.94$_{\pm 2.37}$	27.61$_{\pm 2.13}$	**30.45**$_{\pm 2.40}$
	TLDR	2.40	1.80$_{\pm 0.08}$	2.08	2.15	1.53	2.19$_{\pm 0.29}$	3.37$_{\pm 0.29}$	**3.61**$_{\pm 0.78}$	3.55$_{\pm 0.40}$
	RLHF	2.31	2.05$_{\pm 0.11}$	1.87	2.83	2.57	1.01$_{\pm 0.12}$	1.44$_{\pm 0.07}$	2.82$_{\pm 0.10}$	**3.03**$_{\pm 0.02}$
	HumanEval	47.44	44.30$_{\pm 2.36}$	45.29	46.19	42.22	43.68$_{\pm 1.82}$	47.50$_{\pm 1.57}$	**48.59**$_{\pm 2.08}$	45.10$_{\pm 2.84}$
Mistral-7B	AlpacaEval	33.77	24.99$_{\pm 4.28}$	21.70	28.47	29.31	35.84$_{\pm 0.53}$	41.09$_{\pm 1.56}$	41.43$_{\pm 3.00}$	**47.40**$_{\pm 2.94}$
	TLDR	2.79	3.06$_{\pm 0.24}$	2.90	2.41	3.48	3.28$_{\pm 0.41}$	4.40$_{\pm 0.12}$	**4.80**$_{\pm 0.12}$	4.59$_{\pm 0.20}$
	RLHF	2.56	2.13$_{\pm 0.04}$	1.78	2.88	2.94	1.83$_{\pm 0.15}$	1.70$_{\pm 0.09}$	3.10$_{\pm 0.06}$	**3.42**$_{\pm 0.05}$
	HumanEval	83.63	85.56$_{\pm 1.27}$	84.15	84.09	79.17	82.78$_{\pm 1.25}$	85.24$_{\pm 0.45}$	85.59$_{\pm 1.41}$	**85.67**$_{\pm 0.34}$
Task-aware										
Llama2-7B	RLHF	1.01	1.04$_{\pm 0.04}$	0.66	1.29	1.43	0.97$_{\pm 0.02}$	1.62$_{\pm 0.05}$	**1.69**$_{\pm 0.05}$	1.32$_{\pm 0.05}$
	HumanEval	51.27	51.91$_{\pm 1.61}$	54.74	52.23	53.10	49.85$_{\pm 3.17}$	52.67$_{\pm 0.71}$	**55.09**$_{\pm 1.66}$	50.67$_{\pm 1.24}$
Mistral-7B	RLHF	0.99	1.05$_{\pm 0.04}$	0.56	1.31	1.31	1.15$_{\pm 0.06}$	1.29$_{\pm 0.13}$	**1.71**$_{\pm 0.01}$	1.35$_{\pm 0.07}$
	HumanEval	84.27	83.34$_{\pm 2.54}$	86.75	84.81	79.91	85.51$_{\pm 1.28}$	85.26$_{\pm 1.13}$	**87.35**$_{\pm 1.03}$	84.18$_{\pm 1.63}$

Experimental Results

The Effect of the Number of Monte-Carlo Samples: Positive correlation between performance and generated MC samples



INSTINCT

Use Your INSTINCT: INSTruction optimization for LLMs usIng Neural bandits Coupled with Transformers.

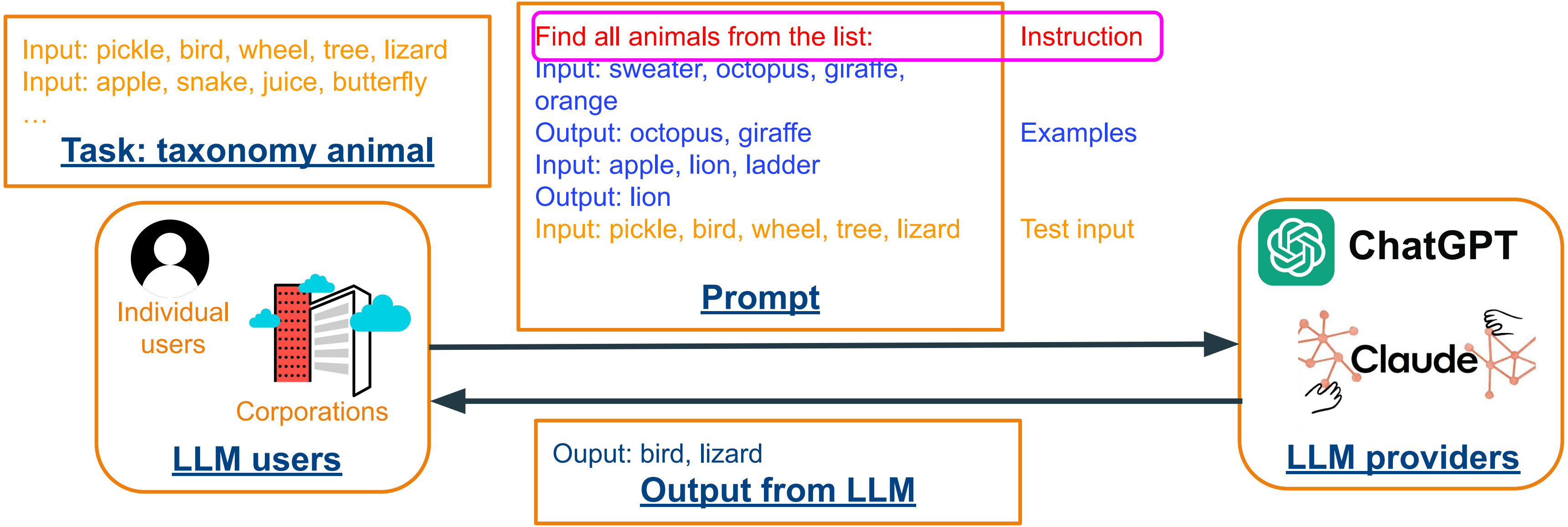
Xiaoqiang Lin*, Zhaoxuan Wu*, Zhongxiang Dai, Wenyang Hu, Yao Shu, See-Kiong Ng, Patrick Jaillet, Bryan Kian Hsiang Low.

ICML 2024.

Motivation

- Few-shot in-context learning

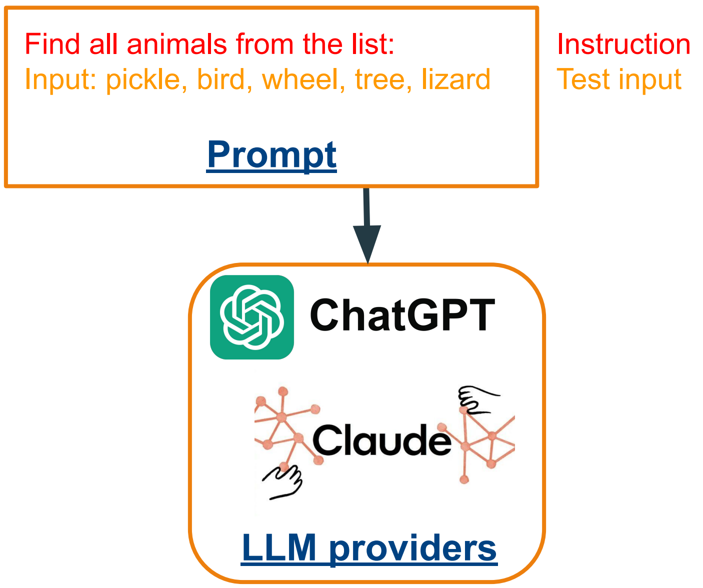
Good instruction is vital to the performance!



Motivation

- Human designed instruction can be costly and not good
- Instruction optimization: Automatically optimize the instructions to obtain the best performance of LLMs

In-context learning



Challenges

- Best performing LLMs are black-box models
  - ChatGPT, Claude
- Access to black-box LLMs is costly
  - API calls are expensive
  - A query-efficient approach is needed

Formulation: Instruction Optimization

- Objective:

$$\rho^* = \mathrm{argmax}_\rho h(\rho)$$

$$h(\rho) := \mathbb{E}_{(x,y)\in D_V} s\big(f(\rho, x), y\big)$$

Preliminary - Bayesian Optimization (BO)

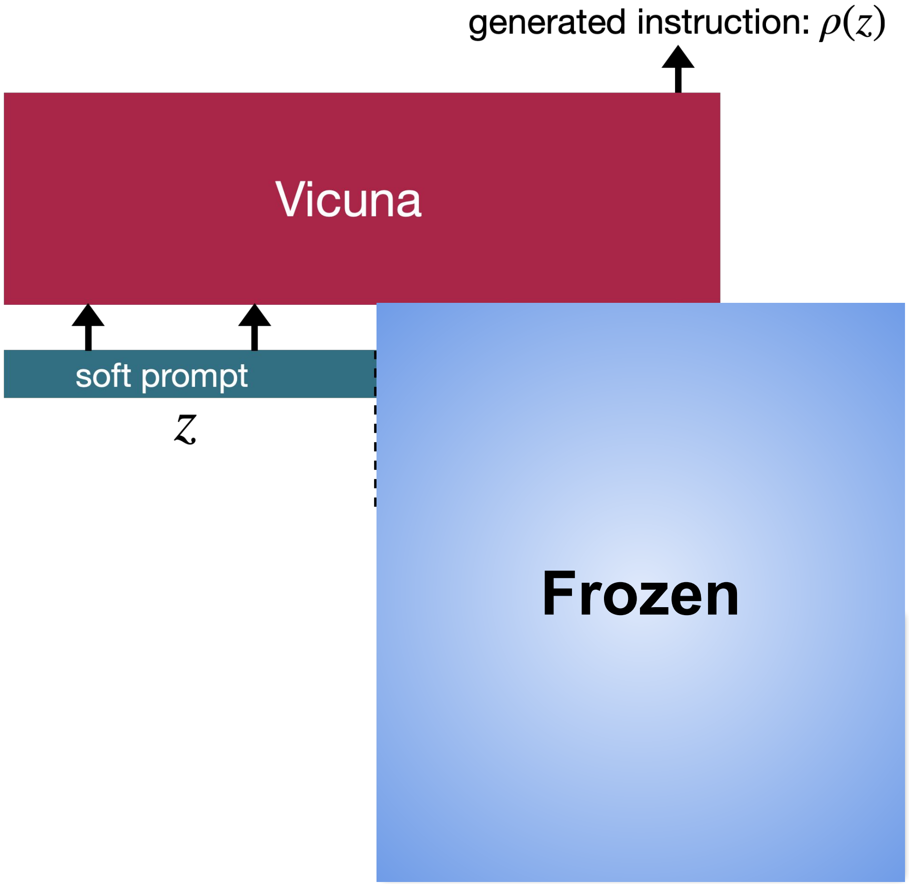
- Sequential black-box optimization: find $\rho^* = \mathrm{argmax}_\rho h(\rho)$
- To choose sequential queries intelligently: $\rho_1, \ldots, \rho_t$
  - Uses a Gaussian process (GP) as a surrogate to model the objective function

Preliminary - Neural Bandits

- Problem with BO:
  - GP is not powerful enough to model the LLM performance.
  - Objective function $h(\rho)$ is not a simple function
- Solution: Use neural networks – neural bandits algorithm
  - Use the neural networks (e.g., transformers) as the surrogate model
  - Can model highly complex functions

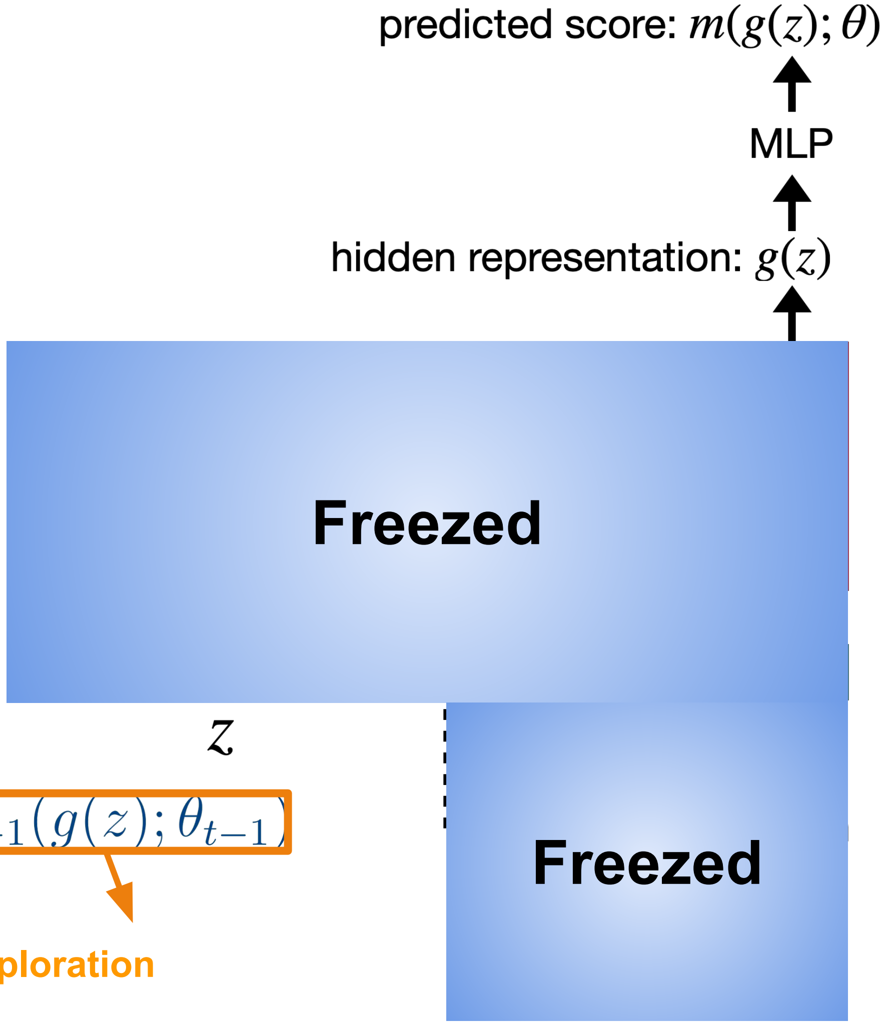
INSTINCT Algorithm

- Map a soft prompt z(a vector in continuous space) into instruction
  - Search in the continuous space $\rho(z)$



INSTINCT Algorithm

- Uses the whole Vicuna as surrogate model to leverage the expressive power of transformer: $m(g(z);\theta)$
- Acquisition function from **NeuralUCB** algorithm:

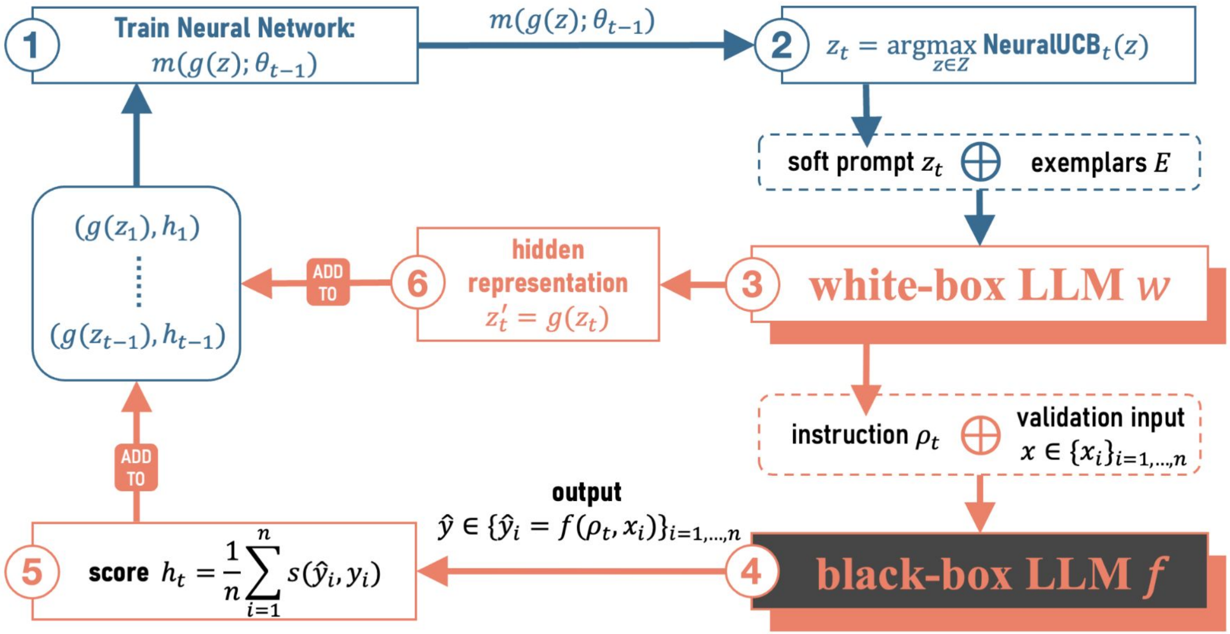


$$z_t = \mathrm{argmax}_{z\in Z}\mathrm{NeuralUCB}_t(z)$$

$$\mathrm{NeuralUCB}_t(z) := \boxed{m(g(z);\theta_{t-1})} + \nu_t\boxed{\sigma_{t-1}(g(z);\theta_{t-1})}$$

Exploitation **Exploration**

INSTINCT Algorithm



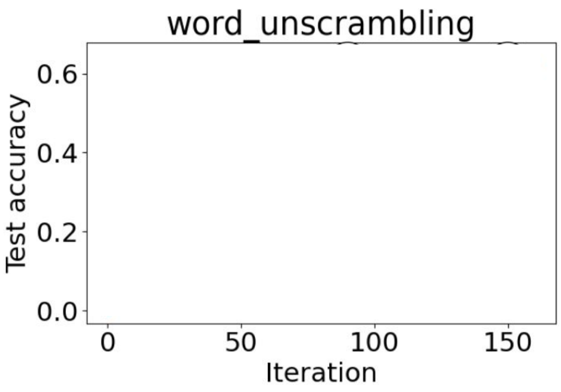
Experiments: Instruction Induction

- Given a task-specific training dataset, find the task-specific instruction that best describes the relationship between inputs and outputs

- Datasets
  - 30 instruction induction tasks curated by [1]

[1] Lichang Chen, Jiuhai Chen, Tom Goldstein, Heng Huang, and Tianyi Zhou. InstructZero: Efficient instruction optimization for black-box large language models. arXiv preprint arXiv:2306.03082, 2023b.

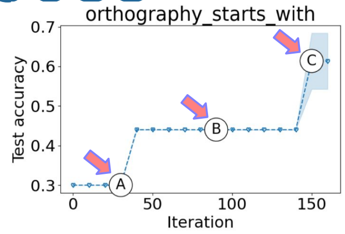
Experiments: Visualizing the Optimization Process



Task description: given a list of shuffled letters, rearrange the letters to form a meaningful word.

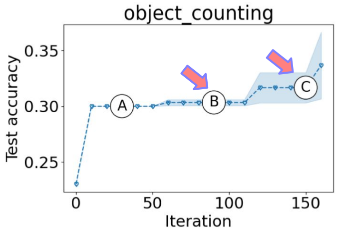
Iteration	Instruction

Experiments: Visualizing the Optimization Process



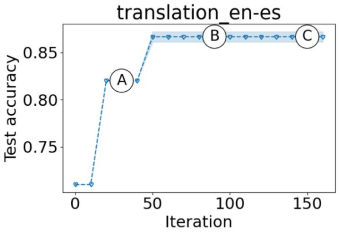
Task description: given a sentence and a letter, output the words that start with the letter in the sentence.

Iteration	Instruction
A	The instruction was to find a word that could be formed by rearranging the letters of the given word
B	The instruction was to find the word that the input corresponds to, and output it
C	The instruction was to output the word that starts with the letter that was inputted



Task description: given a sentence, output the number of objects in the sentence.

Iteration	Instruction
A	The instruction was to output the number of items that the speaker has, given the list of items that the speaker possesses
B	The instruction was to output the number of objects mentioned in the input
C	The instruction was to output the number of items the player has, but the player has entered the number of items instead



Task description: translate the words from English to Spanish.

Iteration	Instruction
A	The instruction was to translate the words from Spanish to English
B	The instruction was to translate the words from English to Spanish
C	The instruction was to translate the words from English to Spanish

Instructions found by INSTINCT improves over iterations

Experiments: Instruction Induction

Task	APE	InstructZero	INSTINCT (ours)
antonyms	0.6367(0.1416)	0.8267(0.0072)	**0.8467(0.0027)**
auto_categorization	0.2500(0.0094)	**0.2567(0.0119)**	0.2500(0.0330)
auto_debugging	0.2917(0.0340)	**0.3750(0.0000)**	0.2917(0.0340)
cause_and_effect	0.5733(0.0891)	**0.8133(0.0109)**	0.5867(0.0871)
common_concept	0.0691(0.0207)	0.0864(0.0398)	**0.2129(0.0019)**
diff	0.6733(0.2667)	0.6933(0.2224)	**1.0000(0.0000)**
informal_to_formal	**0.5736(0.0026)**	0.5310(0.0024)	0.5534(0.0000)
letters_list	**1.0000(0.0000)**	0.5900(0.1674)	**1.0000(0.0000)**
negation	0.7533(0.0109)	0.7767(0.0136)	**0.8167(0.0027)**
object_counting	**0.3633(0.0191)**	0.3600(0.0929)	0.3400(0.0698)
odd_one_out	0.6333(0.0144)	0.6133(0.0871)	**0.7000(0.0163)**
orthography_starts_with	0.4567(0.1477)	0.5067(0.0871)	**0.6667(0.0272)**
rhymes	0.1567(0.0640)	**1.0000(0.0000)**	**1.0000(0.0000)**
second_word_letter	**0.7467(0.2028)**	0.4333(0.1872)	0.1000(0.0411)
sentence_similarity	0.0000(0.0000)	0.0000(0.0000)	**0.1400(0.0047)**
sum	0.6733(0.2667)	**1.0000(0.0000)**	**1.0000(0.0000)**
synonyms	**0.3600(0.0759)**	0.2767(0.0925)	0.3067(0.0491)
taxonomy_animal	0.3467(0.2341)	0.7167(0.0838)	**0.8567(0.0599)**
word_sorting	0.3300(0.0374)	0.3100(0.1143)	**0.5133(0.0027)**
word_unscrambling	0.4400(0.1389)	0.5500(0.0170)	**0.6333(0.0072)**
# best-performing tasks	5	5	13
# second-best-performing tasks	5	10	5
average rank	2.25	2.0	1.45

Experiments: Instruction Induction (Summarization Task)

- INSTINCT also performs the best in another commonly used *SAMSum* benchmark dataset

Method	ROUGE-1	ROUGE-2	ROUGE-L
APE	0.32549	0.10308	0.30245
InstructZero	0.32595	0.10528	0.30061
INSTINCT	**0.35580**	**0.13350**	**0.33600**

Experiments: Improving Zero-shot CoT

- A well-known zero-shot instruction for chain-of-thought (CoT) reasoning form [1] is

"Let's think step by step."

- INSTINCT finds better ones:

Method	Dataset	Best Zero-Shot CoT Instruction	Score
Kojima et al. (2022)	GSM8K	Let's think step by step.	0.71797
InstructZero	GSM8K	Let's use the instruction to solve the problem.	0.74299
INSTINCT (ours)	GSM8K	**Let's think about it.**	**0.74526**
Kojima et al. (2022)	AQUA-RAT	Let's think step by step.	0.52362
InstructZero	AQUA-RAT	Let's break down the problem.	0.54331
INSTINCT (ours)	AQUA-RAT	**I have a new solution.**	**0.54724**
Kojima et al. (2022)	SVAMP	Let's think step by step.	0.7625
InstructZero	SVAMP	Let's use the equation.	0.795
INSTINCT (ours)	SVAMP	**Let's use our brains.**	**0.81**

[1] Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners. In Proc. NeurIPS, 2022.

Conclusion

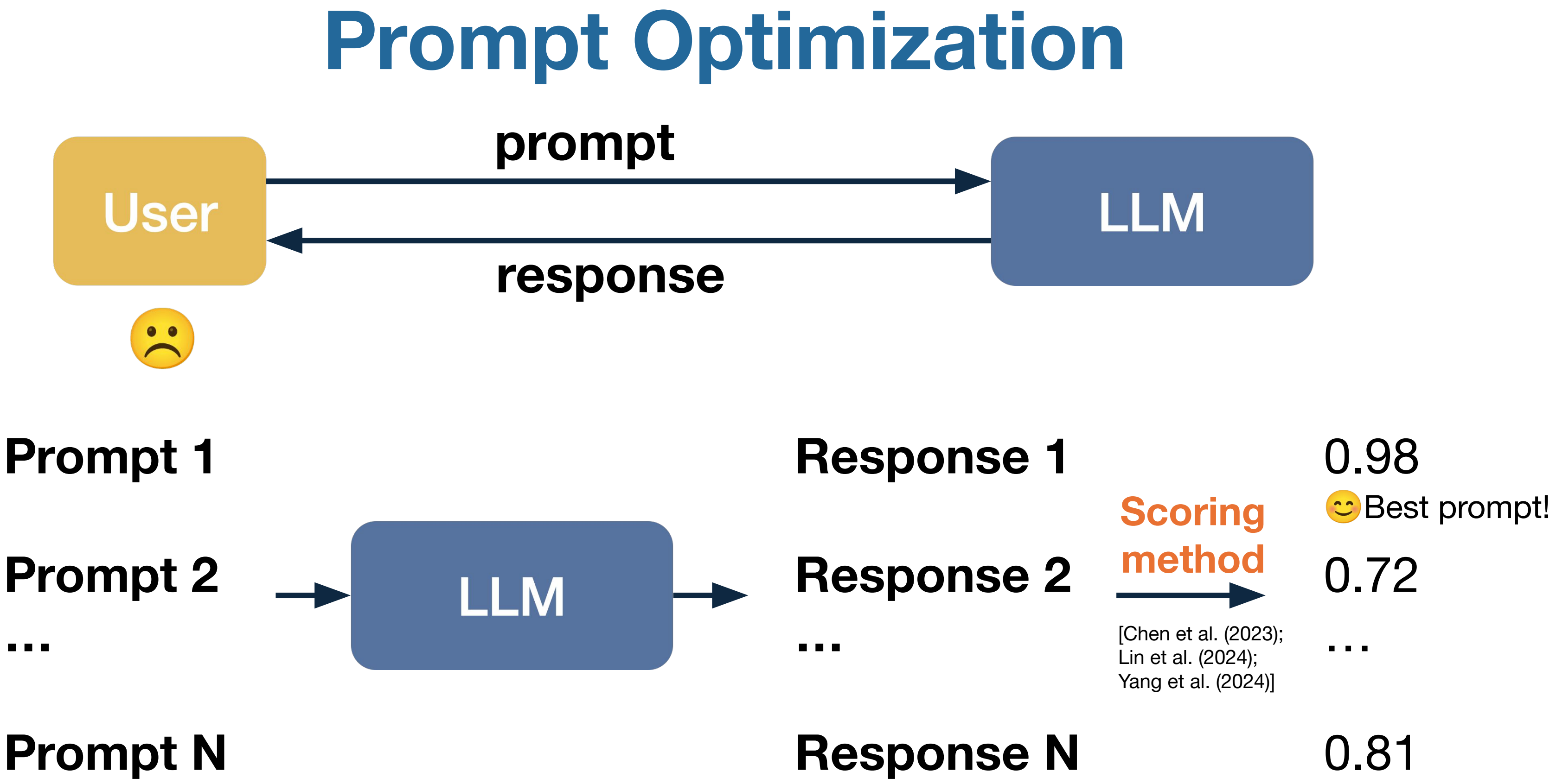
- We introduce the INSTINCT to optimize task-specific instructions for black-box LLMs
- INSTINCT achieves better performance due to the use of neural-bandits algorithm and the expressive power of the transformer.
- We demonstrate on multiple settings that INSTINCT achieves better performance with the same number of queries.

POHF

Prompt Optimization with Human Feedback.

Xiaoqiang Lin, Zhongxiang Dai, Arun Verma, See-Kiong Ng, Patrick Jaillet, Bryan Kian Hsiang Low.

ICML 2024, Workshop on Models of Human Feedback for AI Alignment. Selected as Oral

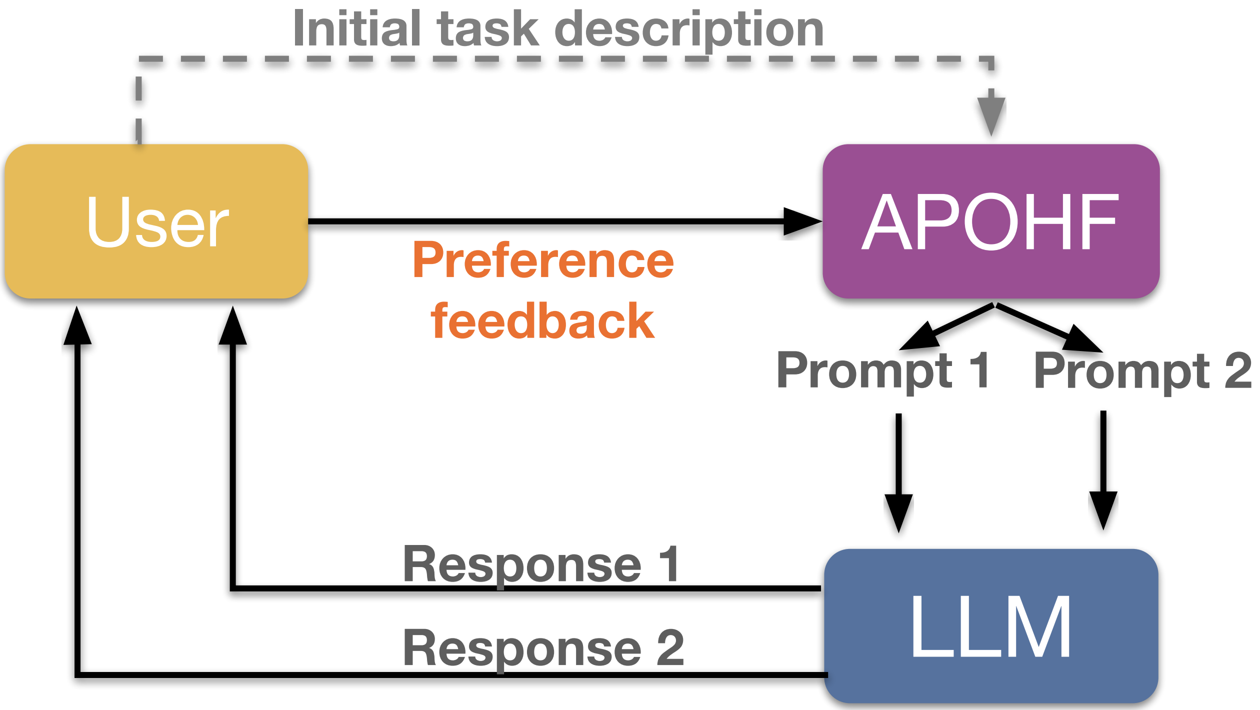

Prompt Optimization
prompt
User
LLM
response
Prompt 1
Prompt 2
...
Prompt N
LLM
Response 1
Response 2
...
Response N
Scoring method
[Chen et al. (2023); Lin et al. (2024); Yang et al. (2024)]
0.98
Best prompt!
0.72
...
0.81

Prompt Optimization

- A scoring method may not be available or reliable
  - No validation dataset available
  - A scorer LLM may not be accurate
  - Human is not good at giving a score (Yue et al. 2012)
- Human is more reliable at providing preference feedback (Yue et al. 2012)

- Can we achieve prompt optimization using only human preference feedback?

Prompt Optimization with Human Feedback



Our algorithm - APOHF

- Using the neural network for latent score prediction
  - $h(x;\theta)$ mapping from prompt to latent score
- Preference feedback model - Bradley-Terry-Luce (BTL) model (Hunter et al. 2004)

$$P(\mathrm{x}_1 \succ x_2) = \sigma(h(x_1;\theta) - h(x_2;\theta))$$

- Given the previous feedback $\boldsymbol{D}_{t-1} = \{\boldsymbol{x}_{s,1}, \boldsymbol{x}_{s,2}, \boldsymbol{y}_s\}_{s=1\,\ldots t-1}$, train the NN ($h$) by minimizing the following loss function:

$$\ell(\theta) = -likelihood\left(y, \sigma\left(h(x_1;\theta) - h(x_2;\theta)\right)\right) + \lambda||\theta||$$

Our algorithm - APOHF

- Selection of first prompt:

$$\mathbf{x}_{t,1} = \arg\max_{x} \boldsymbol{h}(\boldsymbol{x}; \boldsymbol{\theta}_t)$$

- Selection of second prompt:

$$\mathbf{x}_{t,2} = \arg\max_{x} \boxed{\boldsymbol{h}(\boldsymbol{x}; \theta_t)} + \nu \boxed{||\nabla \mathbf{h}(\mathbf{x}; \boldsymbol{\theta}_t) - \nabla \mathbf{h}(\mathbf{x}_{t,1}; \boldsymbol{\theta}_t)||_{V_{t-1}^{-1}}}$$

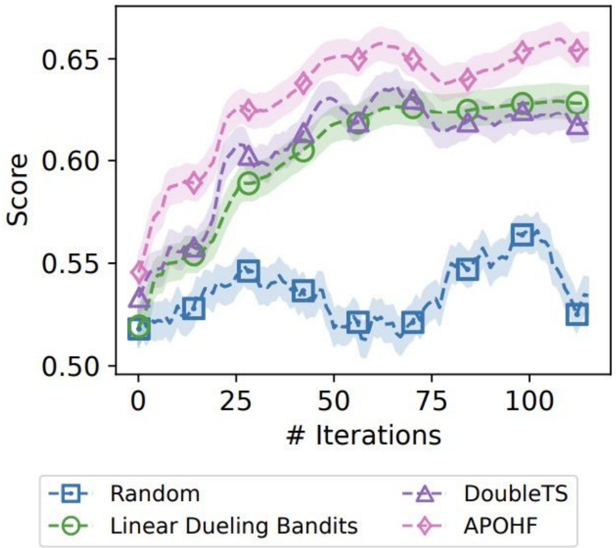
Exploitation: Score prediction

Exploration: Encourages $\mathrm{x}_{t,2}$ to be different from the previously selected prompts and $\mathrm{x}_{t,1}$

(Arun et al. (2025))

Verma, Arun, Zhongxiang Dai, Xiaoqiang Lin, Patrick Jaillet, and Bryan Kian Hsiang Low. "Neural Dueling Bandits." *ICLR 2025*

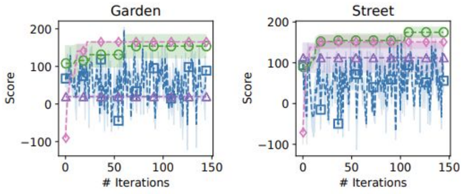
Experiment - Optimization of User Instructions



Task	Iter	Instruction	Score
antonyms	0	add the prefix "un-" to the given words to form their opposites.	0.45
	5	remove the "un-" prefix from each word.	0.45
	10	provide the opposite of the given words.	0.70

Our APOHF consistently and significantly outperforms the other methods.

Experiment - Prompt Optimization for Text-to-Image Generative Models



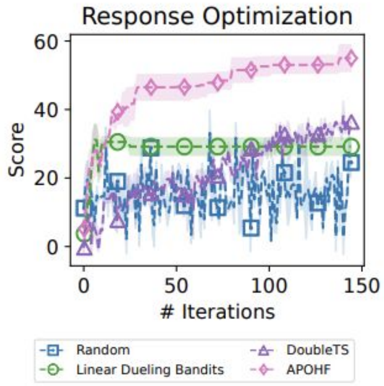
Our APOHF can efficiently produce images which better align with the image the user has in mind.

Experiment - Response Optimization with Human Feedback

The prompt is "Human: What is there to do in Atlantic City?".

Iter	Response	Score
0	There are many activities to do in Atlantic City! You can visit the famous Atlantic City Boardwalk, try your luck at the casinos, enjoy the beach and water activities, shop at the outlets and specialty stores, dine at top-rated restaurants, enjoy live entertainment and nightlife, visit museums and historical sites, and attend events and festivals throughout the year.	81.96
10	There are plenty of things to do in Atlantic City! You can visit the famous boardwalk, try your luck at the casinos, enjoy a show or concert at one of the many entertainment venues, dine at a variety of restaurants, shop at the outlets and specialty stores, relax on the beach, or take a stroll through the historic district. There's something for everyone in Atlantic City!	91.46
20	There are many attractions and activities in Atlantic City, including: 1. Casinos: Atlantic City is known for its many casinos, where you can try your luck at slots, poker, blackjack, and more. 2. Boardwalk: ... 3. Beach: .. 4. Steel Pier: ... 5. Shows and concerts: ... 6. Nightlife: ... 7. Dining: ... 8. Shopping: ...	180.14

Our APOHF is able to further refine the response of an LLM to make it more preferable for human users.



Q & A

- Any questions?

THANK YOU